Norm-based behaviour modification in BDI agents

Felipe Meneguzzi
felipe.meneguzzi@kcl.ac.uk

Michael Luck
michael.luck@kcl.ac.uk
Outline

• Context
• Norms in Agent Languages
• Normative Processing
• Normative AgentSpeak(L)
• Conclusions and Future Work
CONTEXT
Societal Control

• Multiagent systems:
  – Multiple autonomous agents
  – Adaptable to changes
  – Certain degree of unpredictability
  – Require societal control

• Norms provide a valuable mechanism to impose control on agent societies
Normative Systems

• Define standards of acceptable behaviour
• Rely on representation of:
  – Obligations
  – Prohibitions
  – Permissions
• Research largely on the *macro* level
• We need to address how individual agents *adapt* to norms, should they choose to follow them
NORMS IN AGENT LANGUAGES
Norm Representation

• Focuses on the operational aspect of norm compliance by an agent
• Norms are defined in the form:
  $norm(Activation, Expiration, NormCondition)$
• Denoting when a norm becomes active and expires, and what is the object of the norm
Norms and Goal Types

- We narrow norm types down to:
  - Obligations – agent *must* do/achieve something
  - Prohibitions – agent *must not* do/achieve something

<table>
<thead>
<tr>
<th>Norm</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>obligation(p)</td>
<td>add a goal to achieve state p, from <em>Activation</em> to <em>Expiration</em>.</td>
</tr>
<tr>
<td>obligation(a)</td>
<td>add a new plan with a <em>Activation</em> triggering event, and action a in its body.</td>
</tr>
<tr>
<td>prohibition(p)</td>
<td>prevent adoption of plans that bring about state p.</td>
</tr>
<tr>
<td>prohibition(a)</td>
<td>prevent adoption of plans that execute action a.</td>
</tr>
</tbody>
</table>
Norm Perception

• Norms perceived as environmental information
• If accepted, the following flow occurs

Environment / Society → Norms → Accept? → Accept Norm → Verify Consistency → Change Behaviour

Accept?

Reject Norm → Sanctions

Meneguzzi and Luck
Motivating Example

Agent (in AgentSpeak)

+!cleanRoom(Room) : at(Room)
   <- +clean(Room).

+!clean(room1) : true
   <- +at(room1);
      !cleanRoom(room1).

+!clean(classifRoom) : true
   <- +at(classifRoom);
      !cleanRoom(classifRoom).

+cleanClassif : true
   <- !clean(classifRoom).

Norms

\[
\begin{align*}
\text{norm} & (\text{time}(4), \\
& \text{time}(20), \\
& \text{obligation} (\text{clean}(\text{room1}))) \\
\text{norm} & (\text{time}(6), \\
& \text{time}(22), \\
& \text{prohibition} (\text{at} (\text{classifRoom})))
\end{align*}
\]
Expected Result

Events

• time(4)
• time(6)
• cleanClassif
• time(20)
• time(22)

Effects

• Adopt plan to clean room1
• Suppress plan to clean classifRoom
• No plan should be adopted
• Obligation to clean room1 expires
• Plan to clean classifRoom no longer suppressed
In a nutshell

Plan Library

- Plan 1
- Plan 2
- Plan 3
- Plan 4
- Plan 5

Prohibition

Obligation

Plan 6

Meneguzzi and Luck
NORMATIVE PROCESSING
Norm Outcomes

- Generate Plans
  - Plan for Start Condition
  - Plan for End Condition

Accepted Norm

Meneguzzi and Luck
Norm Activation

• Obligations
  – Behaviours associated with obligations must be carried out when they become active
  – Activation condition becomes trigger for plans that achieve obligations

• Prohibitions
  – Behaviours associated with prohibitions must not be carried out when they become active
  – Activation conditions becomes trigger for plans that filter intentions and plan library
Norm Expiration

- When a norm expires, its effects in the plan library must be reversed
- Plans added for obligations can be removed
- Plans suppressed for prohibitions must be restored
# Norm Outcomes

<table>
<thead>
<tr>
<th>Deontic Modality</th>
<th>Activation Condition</th>
<th>Expiration Condition</th>
<th>Outcome</th>
</tr>
</thead>
<tbody>
<tr>
<td>obligation(O)</td>
<td>True</td>
<td>True</td>
<td>Ignore norm</td>
</tr>
<tr>
<td>True</td>
<td>False</td>
<td></td>
<td>Adopt plan to achieve O (if O is a world state) or adopt plan to execute O (if O is an action)</td>
</tr>
<tr>
<td>False</td>
<td>False</td>
<td></td>
<td>Add plan to achieve O (if O is a world state) or add plan that includes O (if O is an action) to PL when activation holds</td>
</tr>
<tr>
<td>False</td>
<td>True</td>
<td></td>
<td>Ignore norm</td>
</tr>
<tr>
<td>prohibition(P)</td>
<td>True</td>
<td>True</td>
<td>Ignore norm</td>
</tr>
<tr>
<td>True</td>
<td>False</td>
<td></td>
<td>Drop intentions to achieve P and suppress plans that achieve P (if P is a world state) or drop intentions that include P and suppress plans that include P (if P is an action)</td>
</tr>
<tr>
<td>False</td>
<td>False</td>
<td></td>
<td>Add plan to suppress plans that achieve P (if P is a world state) or plans that include P (if P is an action) when activation holds</td>
</tr>
<tr>
<td>False</td>
<td>True</td>
<td></td>
<td>Ignore norm</td>
</tr>
</tbody>
</table>
NORMATIVE AGENTSPEAK(L)
Meta-reasoning Operators

- Implementation created using *Jason*
- Extended with meta-level actions:

<table>
<thead>
<tr>
<th>Action</th>
<th>Effect</th>
</tr>
</thead>
<tbody>
<tr>
<td>.plan_steps(P,S)</td>
<td>takes a plan P and unifies its plan steps as a list of literals with S</td>
</tr>
<tr>
<td>.plan_conseq(P,C)</td>
<td>takes a plan P and unifies its declarative consequences with C</td>
</tr>
<tr>
<td>.action(A)</td>
<td>succeeds if A refers to an action</td>
</tr>
<tr>
<td>.literal(L)</td>
<td>succeeds if L to a literal</td>
</tr>
<tr>
<td>.remove_plan(P)</td>
<td>removes P from the plan library</td>
</tr>
<tr>
<td>.suppress_plan(P)</td>
<td>suppresses the specified plan (prevents from being executed)</td>
</tr>
<tr>
<td>.unsuppress_plan(P)</td>
<td>allows a previously suppressed plan to be executed</td>
</tr>
</tbody>
</table>
Plan Modification Strategies

@prohibitionStart(Prohibition)
+!Start : true
  <- !findPlansWithAction(Prohibition, SPlans); !suppressPlans(SPlans);
  +suppressedPlans(Prohibition,SPlans).

@prohibitionEnd(Prohibition)
+!End : suppressedPlans(Prohibition,SPlans)
  <- !unsuppressPlans(SPlans);
  .remove_plan(prohibitionStartStart(Prohibition));
  .remove_plan(prohibitionEndEnd(Prohibition)).
CONCLUSIONS AND FUTURE WORK
Conclusions

• A practical framework for normative processing at the agent level
• Generic enough for application to most modern agent languages
• Working prototype in AgentSpeak(L)
  – www.meneguzzi.eu/felipe/software.html
Future Work

• Current framework is very coarse
• Suppression cannot refer to specific instances of actions/world states
• Future work will allow specific restrictions to be added to norms
Norm-based behaviour modification in BDI agents

Felipe Meneguzzi
Michael Luck
felipe.meneguzzi@kcl.ac.uk